In the Claims:

Please amend Claims 1, 7, 9, 11, 12, 13, 19, 21, 23, 24 and 25, all as shown below. Applicant respectfully reserves the right to prosecute any originally presented or canceled claims in a continuing or future application.

1. (Currently Amended) A system for software source code analysis, comprising:

a plugin framework that includes a plurality of software interfaces to allow software test tools to be plugged into the system in a tool-agnostic manner;

a software interface to a source code management ~~system~~ tool, which can be used to receive information from the source code management tool to identify changes that have ~~occurred~~ been made by developers to source code ~~over~~ during a specified interval of time or during a set of relative edit changes, independent of any particular source code management tool implementation;

a software interface to a code coverage ~~database~~ tool, which can be used to receive information from the code coverage tool to identify ~~what~~ source code that has been exercised by the system during a test run, independent of any particular code coverage tool implementation;

a software interface for identifying which source files are exercised in a software product by a particular software test, using code coverage mapping,

a software interface for identifying ~~what~~ which tests have failed during a test execution cycle for a particular software product, independent of any particular testing technology or test execution tool; and

a bug inspection analyzer that determines ~~the~~ a failure-to-change intersection point by integrating the information ~~culled~~ from the plurality of software interfaces ~~described above~~, including both the software interface to the source code management tool and the software interface to the code coverage tool to determine

~~what~~ which of the tests failed,

~~what~~ which of the source code files of the target software product are exercised by the failing tests as determined by the information received from the code coverage tool, and

which of the files identified thereby have been changed by a ~~product software~~ developer ~~since the failed tests were last known to be in a passing state or within some other specified timeframe~~, during the specified interval of time or the specified set of relative edit changes, as determined by the information received from the source code management tool.

2. (Original) The system of claim 1 wherein changes to the software code between said first version and said second version are identified by change label.

3. (Original) The system of claim 1 wherein changes to the software code between said first version and said second version are identified by modification date.

4. (Original) The system of claim 1 wherein the code coverage interface includes an input for allowing an operator to specify either of date and/or code change ranges to be analyzed by said bug inspection analyzer.

5. (Original) The system of claim 1 wherein the interfaces are realized to interact with a tool-specific implementation that interfaces to a vendor-specific subsystem.

6. (Original) The system of claim 1 wherein the source code interface includes an interface to a vendor-specific SCM system.

7. (Currently Amended) The system of claim 6 wherein ~~wherein~~ the system includes said SCM system.

8. (Original) The system of claim 1 wherein the test interface includes an interface to a TER system.

9. (Currently Amended) The system of claim 8 wherein ~~wherein~~ the system includes said TER system.

10. (Original) The system of claim 1 wherein the test interface includes an interface to a code testing system.

11. (Currently Amended) The system of claim 8 wherein ~~wherein~~ the system includes said code testing system.

12. (Currently Amended) A system for software source code analysis, comprising:

a plugin framework that includes a plurality of software interfaces to allow software test tools to be plugged into the system in a tool-agnostic manner;

means for retrieving a software code and running test suites against it at a first time and at a second time;

means for importing code coverage data into the framework for failure analysis to identify source code that has been exercised by the system during a test run;

means for retrieving detailed set of line-level product changes from a source code management system to identify changes that have been made by developers to source code during a specified interval of time or during a set of relative changes; and,

means for comparing line-level code coverage data for a test case from a code coverage toolset to line-level change information from a source code management system, and determining an intersection of these two data sets to represents the set of critical changes over the specified ~~time period~~ interval of time or specified set of relative edit changes, including determining which of the tests failed, which of the source code files of the target software product are exercised by the failing tests as determined by the information received from the code coverage tool, and which of the files identified thereby have been changed by a developer, during the specified interval of time or the specified set of relative changes, as determined by the information received from the source code management tool.

13. (Currently Amended) A method for software source code analysis, comprising the steps of:

providing a plugin framework that includes a plurality of software interfaces to allow software test tools to be plugged into the system in a tool-agnostic manner, including a software interface for allowing the plugin of source code management tools and a software interface for allowing the plugin of code coverage tools or databases;

accessing a source code management system that is plugged into the plugin framework, which can be used to identify changes that have occurred to source code over a specified interval of time or relative changes, independent of any particular source code management tool implementation;

accessing a code coverage database that is plugged into the plugin framework, which can be used to identify what source code has been exercised during a test run, independent of any particular code coverage tool implementation;

identifying which source files are exercised in a software product by a particular software test, using code coverage mapping;

identifying what tests have failed during a test execution cycle for a particular software product, independent of any particular testing technology or test execution tool; and

determining the failure-to-change intersection point by integrating the information culled from the software interfaces ~~described above~~ to determine what tests failed, what source code files of the target software product are exercised by the failing tests, and which of the files identified thereby have been changed by a product software developer since the failed tests were last known to be in a passing state or within some other specified ~~timeframe~~ interval of time or specified set of relative edit changes.

14. (Original) The system of claim 13 wherein changes to the software code between said first version and said second version are identified by change label.

15. (Original) The system of claim 13 wherein changes to the software code between said first version and said second version are identified by modification date.

16. (Original) The system of claim 13 wherein the code coverage interface includes an input for allowing an operator to specify either of date and/or code change ranges to be analyzed by said bug inspection analyzer.

17. (Original) The system of claim 13 wherein the interfaces are realized to interact with a tool-specific implementation that interfaces to a vendor-specific subsystem.

18. (Original) The system of claim 13 wherein the source code interface includes an interface to a vendor-specific SCM system.

19. (Currently Amended) The system of claim 18 wherein ~~wherein~~ the system includes said SCM system.

20. (Original) The system of claim 13 wherein the test interface includes an interface to a TER system.

21. (Currently Amended) The system of claim 20 wherein ~~wherein~~ the system includes said TER system.

22. (Original) The system of claim 13 wherein the test interface includes an interface to a code testing system.

23. (Currently Amended) The system of claim 20 wherein ~~wherein~~ the system includes said code testing system.

24. (Currently Amended) A method of software source code analysis, comprising the steps of:

providing a plugin framework that includes a plurality of software interfaces to allow software test tools to be plugged into the system in a tool-agnostic manner, including a software interface

for allowing the plugin of source code management tools and a software interface for allowing the plugin of code coverage tools or databases;

retrieving a software code and running test suites against it at a first time and at a second time;

importing code coverage data into the framework for failure analysis;

retrieving detailed set of line-level product changes from a source code management system that is plugged into the plugin framework; and

comparing line-level code coverage data for a test case from a code coverage toolset that is plugged into the plugin framework to line-level change information from a source code management system, and determining an intersection of these two data sets to represents the set of critical changes over the specified ~~time period~~ interval of time or specified set of relative edit changes.

25. (Currently Amended) A computer readable medium including instruction stored thereon which when executed cause the computer to perform the steps of:

providing a plugin framework that includes a plurality of software interfaces to allow software test tools to be plugged into the system in a tool-agnostic manner, including a software interface for allowing the plugin of source code management tools and a software interface for allowing the plugin of code coverage tools or databases;

accessing a source code management system that is plugged into the plugin framework, which can be used to identify changes that have occurred to source code over a specified interval of time or relative changes, independent of any particular source code management tool implementation;

accessing a code coverage database that is plugged into the plugin framework, which can be used to identify what source code has been exercised during a test run, independent of any particular code coverage tool implementation;

identifying which source files are exercised in a software product by a particular software test, using code coverage mapping;

identifying what tests have failed during a test execution cycle for a particular software product, independent of any particular testing technology or test execution tool; and

determining the failure-to-change intersection point by integrating the information culled from the software interfaces ~~described above~~ to determine what tests failed, what source code files of the target software product are exercised by the failing tests, and which of the files identified thereby have been changed by a product software developer since the failed tests were last known to be in a passing state or within some other specified ~~timeframe~~ interval of time or specified set of relative edit changes.